Matching Images to Models – Camera Calibration for 3-D Surface Reconstruction

Robin D. Morris, Vadim N. Smelyanskiy, and Peter C. Cheeseman

NASA Ames Research Center, MS269-2, Moffett Field, CA 94035, USA
[rdm,vadim,cheesem]@ptolemy.arc.nasa.gov

Abstract. In a previous paper we described a system which recursively recovers a super-resolved three dimensional surface model from a set of images of the surface. In that paper we assumed that the camera calibration for each image was known. In this paper we solve two problems. Firstly, if an estimate of the surface is already known, the problem is to calibrate a new image relative to the existing surface model. Secondly, if no surface estimate is available, the relative camera calibration between the images in the set must be estimated. This will allow an initial surface model to be estimated. Results of both types of estimation are given.

1 Introduction

In this paper we discuss the problem of camera calibration, estimating the position and orientation of the camera that recorded a particular image. This can be viewed as a parameter estimation problem, where the parameters are the camera position and orientation. We present two methods of camera calibration, based on two different views of the problem.

1. Using the entire image I, the parameters are estimated by minimizing $(I - \hat{I}(\Theta))^2$, where Θ are the camera parameters and $\hat{I}(\Theta)$ is the image simulated from the (known) surface model.
2. Using features extracted from the image, the parameters are estimated by minimizing $(u - \hat{u}(\Theta))^2$, where $\hat{u}(\Theta)$ is the position of the estimated feature projected into the image plane.

Under the assumption that a surface model is known, the first method has a number of advantages. It makes no assumptions about the size of the displacements between the images; it gives much more accurate estimation as many thousands of pixels are used to estimate a very few camera parameters; most fundamentally for our problem, it does not require feature extraction – images of natural scenes often do not have the sharp corner features required for standard approaches to camera calibration.

In an earlier paper [1] we described a system that inferred the parameters of a high resolution triangular mesh model of a surface from multiple images of that surface. It proceeded by a careful modeling of the image formation process, the process of *rendering*, and showed how the rendering process could be linearized

with respect to the parameters of the mesh (in that case, the height and albedo values). This linearization turned the highly nonlinear optimization for the mesh parameters into the tractable solution of a very high dimensional set of sparse linear equations. These were solved using conjugate gradient, using iterative linearization about the estimate from the previous iteration.

The work in [1] required that the camera parameters (both internal and external) were known, and also assumed that the lighting parameters were known. In this paper we continue to assume that the internal parameters are known – NASA mission sensors are extensively calibrated before launch – and that the lighting parameters are known. Here we will describe how the linearization of the rendering process can be performed with respect to the camera parameters, and hence how the external camera parameters can be estimated by minimizing the error between the observed and synthesized images. We assume the usual pinhole camera model [10].

To estimate the camera parameters as described above requires that a surface model is already available. For the initial set of images of a new region, no surface model is available. In principle one could optimize simultaneously over both the surface parameters and the camera parameters. In practice, because the camera parameters are correlated with *all* the surface parameters, the sparseness of the set of equations is destroyed, and the joint solution becomes computationally infeasible. Instead, for the initial set of images we use the standard approach of feature matching, and minimize the sum squared error of the distance on the image plane of the observed feature and the projection of the estimated feature in 3D.

A surface can be inferred using the camera parameters estimated using feature matching. New images can then be calibrated relative to this surface estimate, and used in the recursive update procedure described in [1].

2 Calibration by Minimizing the Whole Image Error

Consider a surface where the geometry is modeled by a triangular mesh, and an albedo value is associated with each vertex of the mesh. A simulated camera produces an image $\hat{I}$ of the mesh. The camera is modeled as a simple pinhole camera, and its location and orientation is determined by six parameters, its location in space (x_c, y_c, z_c), the intersection of the camera axis with the $x-y$ plane, (x_0, y_0), and the rotation of the camera about the camera axis, ϕ. The last three of these parameters can be replaced by the three camera orientation angles, and we will use both representations in different places, depending on which is more convenient. These parameters are collected into a vector Θ.

For a given surface, given lighting parameters, and known internal camera parameters, the image rendered by the synthetic camera is a function of Θ, ie $\hat{I} = \hat{I}(\Theta)$. Making the usual assumption of independent Gaussian errors, and assuming a uniform prior on Θ, reduces the maximum likelihood estimation

problem to a least-squares problem

$$\hat{\Theta} = \min_{\Theta} \sum_{p} (I_p - \hat{I}_p(\Theta))^2 \tag{1}$$

where $\hat{\Theta}$ is the maximum-likelihood estimate of the camera parameters. Because $\hat{I}(\Theta)$ is in general a nonlinear function of Θ, to make this estimation practical we linearize $\hat{I}(\Theta)$ about the current estimate Θ_0

$$\hat{I}(\Theta) = \hat{I}(\Theta_0) + \mathbf{D}\mathbf{x}; \quad \mathbf{D} = \frac{\partial \hat{I}_p}{\partial \Theta_i} \tag{2}$$

where $\mathbf{D}$ is the matrix of derivatives evaluated at Θ_0, and $\mathbf{x} = \Theta - \Theta_0$. This reduces the least-squares problem in equation 1 to the minimization of a quadratic form, $F_2(\mathbf{x})$,

$$F_2(\mathbf{x}) = \frac{1}{2}\mathbf{x}\mathbf{D}\mathbf{D}^T\mathbf{x}^T - \mathbf{b}\mathbf{x} \tag{3}$$

$$\mathbf{b} = (I - \hat{I}(\Theta))\mathbf{D} \tag{4}$$

which can be solved using conjugate gradient or similar approaches. In the following section we will describe how an *object space* renderer can also be made to compute $\mathbf{D}$, the derivatives of the pixel values with respect to the camera parameters.

2.1 Forming the image

As discussed in [1], to enable a renderer to also compute derivatives it is necessary that all computations are done in object space. This implies that the light from a surface triangle, as it is projected into a pixel, contributes to the brightness of that pixel with a weight proportional to the fraction of the area of the triangle which projects into that pixel. The total brightness of the pixel is thus the sum of the contributions from all the triangles whos projection overlaps with the pixel

$$\hat{I}_p = \sum_{\triangle} f^p_{\triangle} \, \Phi_{\triangle} \tag{5}$$

where $f^p_{\triangle}$ is the fraction of the flux that falls into pixel p, and $\Phi_{\triangle}$ is the total flux from the triangle. This is given by

$$\begin{aligned} &\Phi_{\triangle} = \rho \, E(\alpha^s) \, \cos\alpha^v \, \cos^{\kappa}\theta \, \Delta\Omega, \\ &E(\alpha^s) = \mathcal{A}\,(\mathcal{I}^s \, \cos\alpha^s + \mathcal{I}^a)\,. \\ &\Delta\Omega = S/d^2. \end{aligned} \tag{6}$$

Here ρ is an average albedo of the triangular facet. Orientation angles α^s and α^v are defined in figure 1. $E(\alpha^s)$ is the total radiation flux incident on the triangular facet with area $\mathcal{A}$. This flux is modeled as a sum of two terms. The first

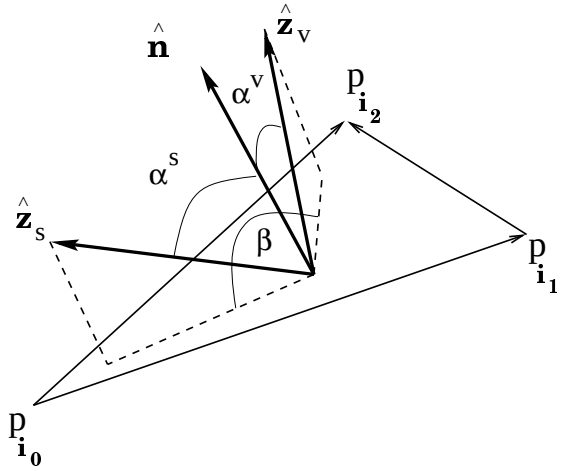

Fig. 1. Geometry of the triangular facet, illumination direction and viewing direction. $\hat{\mathbf{z}}_s$ is the vector to the illumination source; $\hat{\mathbf{z}}_v$ is the viewing direction.

term corresponds to direct radiation with intensity $\mathcal{I}^s$ from the light source at infinity (commonly the sun). The second term corresponds to ambient light with intensity $\mathcal{I}^a$. The parameter θ in equation (6) is the angle between the camera axis and the viewing direction (the vector from the surface to the camera); κ is the lens falloff factor. $\Delta\Omega$ in (6) is the solid angle subtended by the camera which is determined by the area of the lens S and the distance d from the centroid of the triangular facet to the camera. If shadows are present on the surface the situation is somewhat more complex. In this paper we assume that there are no shadows or occlusions present.

2.2 Computing image derivatives with respect to camera parameters

Taking derivatives of the pixel intensities in equation 5 gives

$$\frac{\partial \hat{I}_p}{\partial \Theta_i} = \sum_{\triangle} \left(f^p_{\triangle} \frac{\partial \Phi_{\triangle}}{\partial \Theta_i} + \Phi_{\triangle} \frac{\partial f^p_{\triangle}}{\partial \Theta_i} \right) \tag{7}$$

Consider first $\partial \Phi_{\triangle} / \partial \Theta_i$. We neglect the derivatives with respect to the fall-off angle, as their contribution will be small, and so it is clear from equation 6 that the derivative with respect to any of the camera orientation angles is zero.

The derivative with respect to the camera position parameters is given by

$$\begin{aligned} \frac{\partial \Phi_{\triangle}}{\partial \Theta_i} &\propto \frac{\partial}{\partial \Theta_i} \cos \alpha^v \\ &= \frac{\hat{\mathbf{n}}}{v} (\hat{z}_i - \hat{z}_v (\hat{z}_v . \hat{z}_i)) \end{aligned} \tag{8}$$

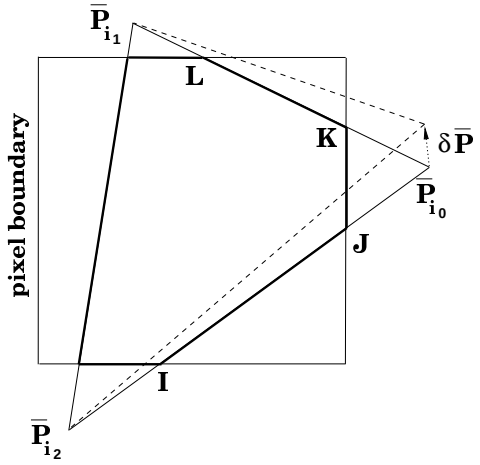

Fig. 2. The intersection of the projection of a triangular surface element $(\mathbf{i_0}, \mathbf{i_1}, \mathbf{i_2})$ onto the pixel plane with the pixel boundaries. Bold lines corresponds to the edges of the polygon resulting from the intersection. Dashed lines correspond to the new positions of the triangle edges when point $\mathbf{P_{i_0}}$ is displaced by $\delta\mathbf{P}$

where $\mathbf{v}$ is the vector from the triangle to the camera, $v = |\mathbf{v}|$, Θ_i are the three components of the camera position, $\hat{z}_i$ are unit vectors in the three coordinate directions and $\hat{z}_v = \mathbf{v}/v$ (see figure 1).

Now consider $\partial f^p_\triangle / \partial \Theta_i$. For triangles that fall completely within a pixel, the second term in equation 7 is zero, as the derivative of the area fraction is zero. For triangles that intersect the pixel boundary, this derivative must be computed. When the camera parameters change, the positions of the projections of the mesh vertices into the image plane will also move. The derivative of the fractional area is given by

$$\frac{\partial f^p_\triangle}{\partial \Theta_i} = \frac{1}{\bar{A}_\triangle} \sum_{\mathbf{j}=\mathbf{i_0},\mathbf{i_1},\mathbf{i_2}} \left(\frac{\partial \bar{A}_{\text{polygon}}}{\partial \bar{\mathbf{P}}_\mathbf{j}} - f^p_\triangle \frac{\partial \bar{A}_\triangle}{\partial \bar{\mathbf{P}}_\mathbf{j}} \right) \frac{\partial \bar{\mathbf{P}}_\mathbf{j}}{\partial \Theta_i}. \tag{9}$$

where $\bar{\mathbf{P}}_\mathbf{j}$ is the projection of point $\mathbf{P_j}$ onto the image plane, and $\bar{A}_\triangle$ is the area of the projection of the triangle. The point displacement derivatives will be detailed below.

Thus, the task of computing the derivative of the area fraction (9) is reduced to the computation of $\partial \bar{A}_\triangle / \partial \bar{\mathbf{P}}_\mathbf{j}$ and $\partial \bar{A}_{\text{polygon}} / \partial \bar{\mathbf{P}}_\mathbf{j}$. Note that the intersection of a triangle and a pixel for a rectangular pixel boundary can, in general, be a polygon with 3 to 7 edges with various possible forms. However the algorithm for computing the polygon area derivatives that we have developed is general, and does not depend on a particular polygon configuration. The main idea of the algorithm can be described as follows. Consider, as an example, the polygon shown in figure 2 which is a part of the projected surface triangle with indices $\mathbf{i_0}, \mathbf{i_1}, \mathbf{i_2}$. We are interested in the derivative of the polygon area with respect to

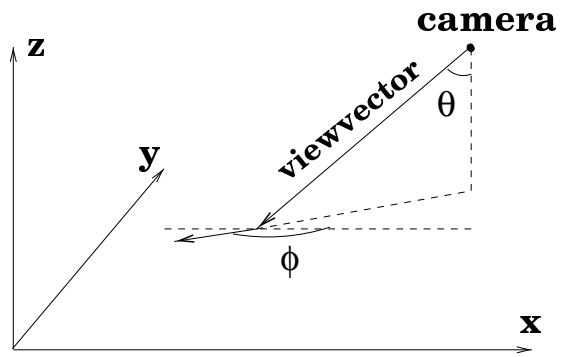

Fig. 3. Illustration of the geometry for determining the rotation between world and camera coordinates.

the point $\bar{\mathbf{P}}_{\mathbf{i_0}}$ that connects two edges of the projected triangle, $(\mathbf{P}_{\mathbf{i_2}}, \mathbf{P}_{\mathbf{i_0}})$ and $(\mathbf{P}_{\mathbf{i_0}}, \mathbf{P}_{\mathbf{i_1}})$. These triangular edges contain segments ($\mathbf{I}$, $\mathbf{J}$) and ($\mathbf{K}$, $\mathbf{L}$) that are sides of the corresponding polygon. It can be seen from figure 2 that when the point $\bar{\mathbf{P}}_{\mathbf{i_0}}$ is displaced by $\delta\bar{\mathbf{P}}_{\mathbf{i_0}}$ the change in the polygon area is given by the sum of two terms

$$\delta\bar{A}_{\text{polygon}} = \delta A_{\mathbf{I},\mathbf{J}} + \delta A_{\mathbf{K},\mathbf{L}}$$

These terms are equal to the areas spanned by the two corresponding segments taken with appropriate signs. Therefore the polygon area derivative with respect to the triangle vertex $\bar{\mathbf{P}}_{\mathbf{i_0}}$ is represented as a sum of the two "segment area" derivatives for the 2 segments adjacent to a given vertex. The details of this computation will be given elsewhere.

We now consider the derivatives of the point positions.

Derivatives of the position of the projection of a point on the image plane. The pinhole camera model gives

$$\hat{u}_l = \frac{[\mathbf{AR}(\mathbf{P}-\mathbf{t})]_l}{[\mathbf{R}(\mathbf{P}-\mathbf{t})]_3} \tag{10}$$

where $\mathbf{R}$ is the rotation matrix from world to camera coordinates, $\mathbf{t}$ is the translation of between camera and world coordinates and $\mathbf{A}$ is the matrix of camera internal parameters [13]. In the numerical experiments presented here we assume that the internal camera parameters are known, and further that the image plane axes are perpendicular, and that the principle point is at the origin. This reduces $\mathbf{A}$ to a diagonal matrix with elements $(k_1, k_2, 1)$, where $k_1 = -\mathrm{f}/l_x$, $k_2 = -\mathrm{f}/l_y$. Where f is the focal length of the lens and l_x and l_y are the dimensions of the pixels in the retinal plane.

The rotation matrix $\mathbf{R}$ can be written in terms of the *Rodrigues* vector [10] $\varrho = (\varrho_1, \varrho_2, \varrho_3)$ which defines the axis of rotation, and $\theta = |\varrho|$ is the magnitude of the rotation. (Clearly ϱ can be written in terms of the camera position, the

look-at point and the view-up vector.)

$$\mathbf{R} = \mathbf{I} - \mathcal{H}\frac{\sin\theta}{\theta} + \mathcal{H}^2\frac{(1-\cos\theta)}{\theta^2} \tag{11}$$

where

$$\mathcal{H} = \begin{pmatrix} 0 & -\varrho_3 & \varrho_2 \\ \varrho_3 & 0 & -\varrho_1 \\ -\varrho_2 & \varrho_1 & 0 \end{pmatrix}. \tag{12}$$

Let $H = \mathcal{H}/\theta$ and $r_i = r_i/\theta$ then

$$\begin{aligned}\frac{\partial \mathbf{R}}{\partial \varrho_i} = &-\mathcal{H}_i\frac{\sin\theta}{\theta} + (H\mathcal{H}_i + \mathcal{H}_i H)\frac{(1-\cos\theta)}{\theta} - H\left(\cos\theta - \frac{\sin\theta}{\theta}\right) r_i \\ &+ H^2\left(\sin\theta - 2\frac{1-\cos\theta}{\theta}\right) r_i\end{aligned} \tag{13}$$

where $\mathcal{H}_i = \partial\mathcal{H}/\partial\varrho_i$. Then

$$\frac{\partial \hat{u}_l}{\partial \varrho_i} = \left(\frac{\left[\mathbf{A}\frac{\partial \mathbf{R}}{\partial \varrho_i}(\mathbf{P}-\mathbf{t})\right]_l}{[\mathbf{R}(\mathbf{P}-\mathbf{t})]_3} - \frac{[\mathbf{AR}(\mathbf{P}-\mathbf{t})]_l\left[\frac{\partial \mathbf{R}}{\partial \varrho_i}(\mathbf{P}-\mathbf{t})\right]_3}{([\mathbf{R}(\mathbf{P}-\mathbf{t})]_3)^2}\right) \tag{14}$$

The derivatives with respect to the position parameters are

$$\frac{\partial \hat{u}_l}{\partial \mathbf{t}_j} = \frac{[\mathbf{AR}(\mathbf{P}-\mathbf{t})]_l[\mathbf{R}]_{3,j}}{([\mathbf{R}(\mathbf{P}-\mathbf{t})]_3)^2} - \frac{[\mathbf{AR}]_{l,j}}{[\mathbf{R}(\mathbf{P}-\mathbf{t})]_3} \tag{15}$$

In practice, optimization using the camera orientation angles directly is inadvisable, as a small change in the angle can move the surface a long distance in the image, and because the minimization in equation 1 is based on a sum over all pixels in the image, this can make for rapid changes in the cost, and failure to converge. Instead we use the "look-at" point (x_0, y_0) which is in the natural length units of the problem. The conversion of the derivatives from angles to look-at is an application of the chain rule, and is not detailed here.

We now consider the second problem, calibration using features detected in the images.

3 Calibration by Minimizing Feature Matching Error

It is well known that camera calibration can be performed using corresponding features in two or more images [4]. This estimation procedure also returns the 3D positions of the corresponding image features. So the parameter space is augmented from Θ^f (where f indexes the frame, or camera parameter set) with $\mathbf{P}_n$, the positions of the 3D points.

If it is assumed that the error between $\mathbf{u}_n^f$, the feature located in image f corresponding to 3D point $\mathbf{P}_n$, and $\hat{\mathbf{u}}_n^f$, the projection of $\mathbf{P}_n$ into image f using

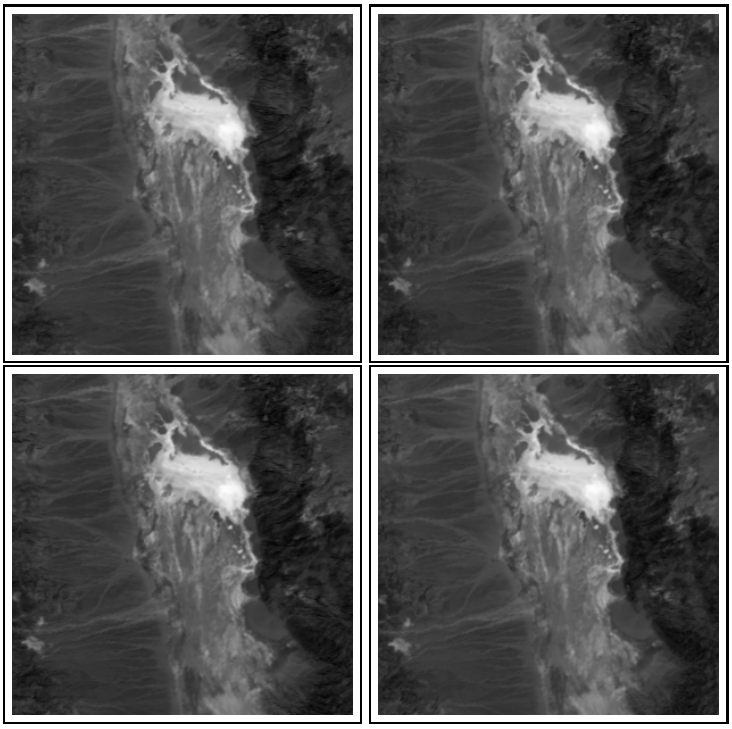

Fig. 4. Four synthetic images of an area of Death Valley

camera parameters Θ^f, is normally distributed, then the negative log-likelihood for estimating Θ and $\mathbf{P}$ is

$$L(\Theta, \mathbf{P}) = \sum_{f=1}^{K} \sum_{n \in \Omega_f} \sum_{l=1}^{2} (u_{l,n}^f - \hat{u}_{l,n}^f)^2 \tag{16}$$

$$\mathbf{P} = \{\mathbf{P}_n;\, n = 1 \ldots N\} \tag{17}$$

$$\Theta = \{\Theta_i^f : \, i = 1 \ldots 6; \quad f = 1 \ldots K\} \tag{18}$$

where Ω_f is the set of features that are detected in image f. Note that the features detected in a given image may well be a subset of all the $\mathbf{P}_n$'s. l indexes the components of $\mathbf{u}$. This form of the likelihood assumes that there is no error in the location of the features in the images.

Typically the non-linear likelihood in equation 18 is minimized using a standard non-linear minimization routine, for example the Levenberg-Marquardt algorithm [11]. The dimensionality of the parameter space in equation 18 is large, equal to 6× (the number of images -1) + 3× the number of 3D points -1, where the parameters of the reference camera are not included, and the overall spatial scale is arbitrary. In general there will be many points, because of this large dimensionality it is important to use exact derivatives, to avoid slow convergence

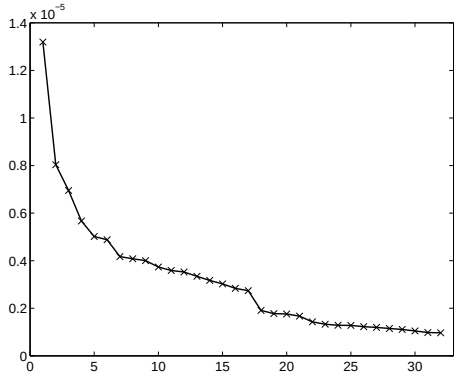

Fig. 5. "Strength" of the features found in image 0

and reduce the need for good initialization. In section 3.2 below we derive the analytic derivatives of the likelihood function, enabling this robust convergence.

The other practical problem in using feature-based camera calibration is the detection and matching of image features.

3.1 Robust feature matching

The maximum likelihood solution to the camera parameter estimation problem is known to be *extremely* sensitive both to mismatches in the feature correspondences, and to even small errors in the localization of the detected features. To reliably estimate the camera parameters we need *reliably located features, reliably matched.* More accurate estimation results from using a smaller set of well localized and well matched features, than a much larger set that includes even a single outlier. Extreme conservatism in both feature detection and matching is needed.

The feature detector most commonly used is the Harris corner detector [2]. This feature detector was developed in the context of images of man-made environments, which contain many strong corners. Remote sensed images of natural scenes of the type we are concerned with (see figure 4) contain some, but much fewer, strong corner features. If the "feature strength" given by the Harris detector is plotted for the features detected on the images in figure 4, it can be seen to fall off rapidly – see figure 5. For this type of image, it is therefore necessary to use only a small number of features, where the associated feature strength is high enough to ensure accurate feature localization. This is a limiting factor in the use of feature based calibration for this type of images, and a motivation for developing the whole image approach described above. Feature detectors more suited to natural scenes are clearly needed, but there will always be particularly mute environments where feature based methods will fail.

Because of the extreme sensitivity to mismatches, it it necessary to ensure that the features found are matched reliably. This is a classic chicken-and-egg

problem: to determine reliable matches it is necessary to correctly estimate the camera parameters, and to correctly estimate the camera parameters requires reliable matches. For this reason, feature matching has spawned a number of methods based on *robust estimation* (and one approach which attempts to do away with explicit feature matching altogether [12]).

The RANSAC algorithm [3] finds the largest set of points consistent with a solution based on a minimal set, repeatedly generating trial minimal sets until the concensus set is large enough. Zhang [4] also bases his algorithm on estimation using minimal sets, but uses LMedS (Least Median Squares) to select the optimal minimal set. The estimate of the fundamental matrix [10] generated using that minimal set is used to reject outliers. Zhang also applies a relaxation scheme to disambiguate potential matches. This is a formalization of the heuristic that features nearby in one image are likely be close together in another image, and in the same relative orientation. In our work we use a modification of Zhang's algorithm described below. Torr and co-workers have developed MLESAC [7] and IMPSAC [6] as improvements on RANSAC. IMPSAC uses a multiresolution framework, and propagates the probability density of the camera parameters between levels using importance sampling. This achieves excellent results, but was considered excessive for our application, where prior knowledge of the types of camera motion between frames is known.

Our algorithm proceeds as follows:

1. Use the Harris corner detector to identify features, rejecting those which have feature strength too low to be considered reliable.
2. Generate potential matches using the normalized correlation score between windows centered at each feature point. Use a high threshold (we use $t = 0.9$) to limit the number of incorrect matches.
3. Use LMedS to obtain a robust estimate of the fundamental matrix:
   - Generate an 8 point subsample from the set of potential matches, where the 8 points are selected to be widely dispersed in the image (see [4]).
   - Estimate the fundamental matrix. Zhang uses a nonlinear optimization based approach. We have found that the simple eight point algorithm [9] is suitable, because our features are in image plane coordinates, which correspond closely to the normalization suggested in [8].
   - Compute the residuals, $\mathbf{u}^T\mathbf{F}\mathbf{u}$ for all the potential matches, and store the *median* residual value.
   - Repeat for a new subsample. The number of subsamples required to ensure with a sufficiently high probability that a subset with no outliers has been generated depends on the number of features and the estimated probability that each potential match is an outlier.
   - Identify $\mathbf{F}_{\min}$ as the fundamental matrix which resulted in the lowest median residual value.
4. Use $\mathbf{F}_{\min}$ to reject outliers by eliminating matches which have residuals greater than a threshold value. (We used $t_{\min} = 1.0$ pixels.)
5. Use the following heuristic to eliminate any remaining outliers: because the images are remote sensed images, the variations in heights on the surface

are small in comparison to the distance to the camera. So points on the surface that are close together should move similar amounts, and in similar directions[1]. The heuristic is

- Consider all features within a radius $r = 0.2$ of the image size from the current feature. The match found for that feature is accepted if both of the following conditions hold:
  (a) The length of the vector between the features is less than a threshold times the length of the largest vector between features in the neighbourhood.
  (b) The average distance between neighbouring features in one image is less than a threshold times the average distance between the same features in th second image.
  In both cases the threshold used was 1.3.

Features are matches between all pairs of images in the set, and are used in the likelihood minimization (18) to estimate the camera parameters. Note that not all features will be detected in all the images in a set, so the likelihood will only contain terms for the features actually found in that image.

3.2 Computing derivatives of the feature positions

To effectively minimize $L(\Theta, \mathbf{P})$ in equation 18 we need to compute its derivatives, which reduces to computing $\frac{\partial \hat{u}^f_{l,n}}{\partial \Theta^f_i}$ and $\frac{\partial \hat{u}^f_{l,n}}{\partial \mathbf{P}_n}$. In what follows we will concentrate on one frame and drop the f index.

We have already shown in equation 14 the expression for the derivative of the point position with respect to the rotation angles and camera position, which together make up $\frac{\partial \hat{u}^f_{l,n}}{\partial \Theta^f_i}$. It remains only to give the expression for the derivative with respect to the 3D feature point. This is the same as the derivative with respect to the camera position, see equation 15, but with the sign reversed, giving

$$\frac{\partial \hat{u}_{l,n}}{\partial \mathbf{P}_{n,j}} = -\frac{[\mathbf{AR}(\mathbf{P}_n - \mathbf{t})]_l [\mathbf{R}]_{3,j}}{([\mathbf{R}(\mathbf{P}_n - \mathbf{t})]_3)^2} + \frac{[\mathbf{AR}]_{l,j}}{[\mathbf{R}(\mathbf{P}_n - \mathbf{t})]_3} \tag{19}$$

Where the subscript n indexes the features.

4 Results and conclusions

Figure 4 shows four synthetic images of Death Valley. The images were generated by rendering a surface model from four different viewpoints and with different lighting parameters. The surface model was generated by using the USGS Digital Elevation Model of the area for the heights, and using scaled intensities of a

[1] This is not true if the camera moves towards the surface, but even then, if the movement towards the surface is not excessive, this heuristic still approximately holds.

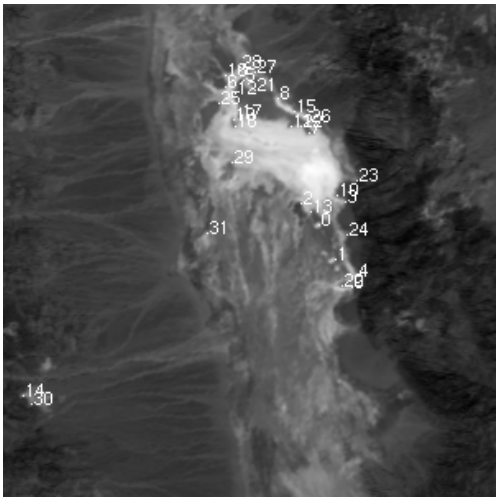

Fig. 6. Features found in image 0

LANDSAT image as surrogate albedos. The size of the surface was approximately 350×350 points, and the distance between grid points was taken as 1 unit. The images look extremely realistic.

Table 1 shows the results of estimating the camera parameters using features detected in the images. These estimates are good, but far from exact. Considering the images in figure 4, it is clear that there are few strong corner features. Figure 6 shows the set of strong features detected in image 0 (the top left image in figure 4). Two things are apparent. Firstly, the features are all due to rapid changes in albedo. Secondly, with two exceptions, the features are clustered. This clustering reduces the accuracy of the estimation. That the features are mostly albedo features confirms that feature based approaches are not applicable to many of the types of image we are interested in.

Table 1 also shows the results for calibration using matching to a pre-existing 3D surface model. The estimation was initialized at the results of the point-matching estimation. The minimization of $(I - \hat{I}(\Theta))^2$ was performed iteratively, re-rendering to compute a new $\hat{I}$ at the value of Θ at the convergence of the previous minimization. As expected, the estimates are very significantly better than the results from point matching, and are very accurate. However, these results are predicated on the existence of a surface model.

These results suggest an approach to camera calibration that is the subject of our current, ongoing, research. Point matching can be used to estimate initial camera parameters, and a very sparse surface representation. A dense surface (shape and albedo) can then be inferred using these camera parameters (see [1]). The whole-image matching approach can be used to re-estimate the camera parameters, and a new surface estimate can be made using the new camera pa-

		true	point-match estimate	whole-image estimate
image 1	camera	$(700, 1416, 4000)$	$(610, 1410, 4030)$	$(685, 1409, 4001)$
	look at	$(205, 1416, 0)$	$205, 1420, 0)$	$(205, 1416, 0)$
	view up	$(0, 1, 0)$	$(-0.005, 1, 0, 002)$	$(0, 1.0, 0.002)$
image 2	camera	$(200, 900, 4000)$	$(200, 968, 4050)$	$(203, 894, 3996)$
	look at	$(205, 1416, 0)$	$(206, 1410, 0)$	$(205, 1416, 0)$
	view up	$(0, 1, 0)$	$(-0.015, 0.994, 0.11)$	$(0, 0.993, 0.129)$
image 3	camera	$(200, 1900, 4000)$	$(176, 1780, 4030)$	$(196, 1881, 4001)$
	look at	$(205, 1416, 0)$	$(206, 1420, 0)$	$(205, 1416, 0)$
	view up	$(0, 1, 0)$	$(-0.007, 0.996, -0.090)$	$(0, 0.993, -0.116)$

Table 1. Results for camera parameter inference. Image 0 was the reference image with parameters camera – $(-300, 1416, 4000)$, look at – $(205, 1416, 0)$ and view up – $(0, 1, 0)$

.

rameter estimates. This process can be iterated. The convergence of this iterative procedure is currently being studied.

References

1. Smelyanskiy, V.N., Cheeseman, P., Maluf, D.A. and Morris, R.D.: Bayesian Super-Resolved Surface Reconstruction from Images. Proceedings of the International Conference on Computer Vision and Pattern Recognition, Hilton Head, 2000
2. Harris, C.: A Combined Corner and Edge Detector. Proceedings of the Alvey Vision Conference, pp 189-192, 1987
3. Fischler, M.A. and Bolles, R.C.: Random Sample Concensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the ACM, June 1981, vol. 24, no. 6, pp 381-395
4. Zhang, Z., Deriche, R., Faugeras, O. and Luong, Q.T.: A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry. AI Journal, vol. 78, pp 87-119, 1994
5. Rousseeuw, P.J.: Robust Regression and Outlier Detection. Wiley, New York, 1987
6. Torr, P.H.S. and Davidson, C.: IMPSAC: Synthesis of Importance Sampling and Random Sample Consensus. Technical Report, Microsoft Research, Cambridge, UK
7. Torr, P.H.S. and Zisserman, A.: MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. Computer Vision and Image Understanding, vol 1, pp 138-156, 2000
8. Hartley, R.I.: In Defense of the Eight Point Algorithm IEEE Transactions on Pattern Aalysis and Machine Intelligence, vol 19, pp 580-594, June 1997
9. Longuet-Higgins, H.C.: A Computer Algorithm for Reconstructing a Scene from Two Projections. Nature, vol 293, pp 133-135, 1981
10. Faugeras, O.: Three-Dimensional Computer Vision MIT Press, 1993
11. More, J.: The Levenberg-Marquardt Algorithm, Implementation and Theory. In G. A. Watson, editor, Numerical Analysis, Lecture Notes in Mathematics 630. Springer-Verlag, 1977.

12. Dellaert, F., Seitz, S.M., Thorpe, C.E., and Thrun, S.: Structure from Motion Without Correspondences. Proceedings of the International Conference on Computer Vision and Pattern Recognition, Hilton Head, 2000
13. Zhang, Z.: A Flexible New Technique for Camera Calibration. Technical Report MSR-TR-98-71, Microsoft Research, Redmond, Washington